MANUAL DE USUARIO

1. PROCESOS DEL MÓDULO ESTADÍSTICO

1.1 Flujo de procesos de ventanas

1.1.1 Consulta de Estadísticas Ventana Principal

Nombre: Estadísticas

Actores: Profesores - Administradores

Función: Mostrar las opciones de reportes a generar.

Descripción: El usuario podrá conocer, a través de los gráficos estadísticos, la situación en un período de tiempo determinado de profesores, alumnos y materias.

1.1.2 Formatos de eventos

Evento: Presentación de pantalla de consulta

1. Escoge la opción del menú
2. Carga de ventana de consulta solicitada

1.2.1 Consulta de Estadísticas de profesores

Nombre: Profesores por materia

Actores: Profesores - Administradores

Función: Presentar la situación de los profesores considerando el rendimiento de los alumnos en cada una de las materias que imparten.

Descripción: El usuario podrá visualizar todos los profesores y las materias que tienen asignadas, así como también consultará por el nombre de los docentes, identificación o fecha de ingreso a la institución.

1.2.2 Formatos de eventos

Evento: Presentación de profesores y materias asignadas

1. Escoger la forma de cómo se desea mostrar la información, todos, por nombre, identificación, o fecha de ingreso.
2. Ejecución de sentencias SQL para mostrar datos en pantalla de acuerdo al requerimiento.
3. Listado de profesores y las materias asignadas a cada uno.
4. Escoger una materia de un profesor, o escoger el profesor para ver la situación de todas sus materias.
5. Ejecución de los Bean de Datos: mostrará el gráfico dependiendo de la opción escogida.

1.3.1 Consulta de Estadísticas de evaluación de profesores

Nombre: Evaluación de profesores.

Actores: Profesores - Administradores

Función: Presentar la calificación de los profesores considerando las evaluaciones realizadas a los alumnos en cada una de las materias que imparten.

Descripción: El usuario podrá visualizar la calificación luego de las evaluaciones que se realizan a los alumnos sobre la calidad, capacidad y comportamiento de los docentes en las materias que tienen asignadas. El gráfico mostrará la calificación de 5 categorías, excelente, muy bueno, bueno regular y malo.

1.3.2 Formatos de eventos

Evento: Presentación de evaluación de profesores.

1. Escoger la forma de cómo se desea mostrar la información, todos, por nombre, identificación, o fecha de ingreso.
2. Ejecución de sentencias SQL para mostrar datos en pantalla de acuerdo al requerimiento.
3. Listado de profesores y las materias asignadas a cada uno.
4. Escoger una materia de un profesor, o escoger el profesor para ver la situación de todas sus materias.

5. Ejecución de los Bean de Datos: mostrará el gráfico dependiendo de la opción escogida.

1.4.1 Consulta de Estadísticas de materias

Nombre: Notas por materia.

Actores: Profesores - Administradores

Función: Presenta el rendimiento de los alumnos en la materia o materias presentadas.

Descripción: El usuario podrá visualizar el rendimiento de los alumnos en una materia específica, lo que permitirá conocer el número de alumnos aprobados, reprobados y aquellos que van a recuperación.

1.4.2 Formatos de eventos

Evento: Presentación de Notas por Materias.

1. Escoger la forma de cómo se desea mostrar la información, todas, o por nombre.
2. Ejecución de sentencias SQL para mostrar datos en pantalla de acuerdo al requerimiento.
3. Listado de las materias.
4. Escoger una materia.

Ejecución de los Bean de Datos: mostrará el gráfico dependiendo de la materia escogida.

1.5.1 Consulta de Estadísticas de alumnos

Nombre: Estadísticas por alumnos.

Actores: Profesores - Administradores

Función: Presenta el rendimiento de los alumnos en la materia o materias presentadas.

Descripción: El usuario podrá visualizar el rendimiento de los alumnos en las materias que cursa durante un período de clases, y su comportamiento en períodos anteriores.

1.5.2 Formatos de eventos

Evento: Presentación de Estadísticas por alumnos.

5. Escoger la forma de cómo se desea mostrar la información, todos, por nombre o identificación.
6. Ejecución de sentencias SQL para mostrar datos en pantalla de acuerdo al requerimiento.
7. Listado de alumno(s).
8. Escoger un alumno.

Ejecución de los Bean de Datos: mostrará el gráfico dependiendo del alumno escogida.

2. DESCRIPCION DE PROGRAMAS

2.1 Descripción de programas de consultas

2.1.1 LISTAR_PROFESORES.JSP

Descripción: Permitirá al usuario mostrar información de profesores dependiendo de la materia(s) que escoja; podrá mostrar todos los profesores, por el nombre o la identificación.

Enlaces: MostrarEstadisticas, EstadisticasAction, struts-config.xml

Librerías: Taglib – Mensajes - Bean – Html

2.1.2 LISTAR_EVALUACIONES.JSP

Descripción: Permitirá al usuario mostrar información de las evaluaciones realizadas por alumnos midiendo la asimilación de conocimientos brindada por el docente en cada una de las materias que imparten..

Enlaces: MostrarEstadisticas, EstadisticasAction, struts-config.xml

Librerías: Taglib – Mensajes - Bean – Html

2.1.3 LISTAR_ALUMNOS.JSP

Descripción: El usuario podrá visualizar el rendimiento de los alumnos en un período determinado, considerando las materias que esté cursando.

Enlaces: MostrarEstadisticas, EstadisticasAction, struts-config.xml

Librerías: Taglib – Mensajes - Bean – Html

2.1.4 LISTAR_MATERIAS.JSP

Descripción: Permite visualizar la información de las materias y conocer el comportamiento de los alumnos que la estén cursando.

Enlaces: MostrarEstadisticas, EstadisticasAction, struts-config.xml

Librerías: Taglib – Mensajes - Bean – Html

2.1.5 MOSTRAR_ESTADISTICAS.JSP

Descripción: Encargada de la presentación de gráficos dependiendo del criterio escogido por el usuario.

Enlaces: EstadisticasAction, struts-config.xml

Librerías: Taglib – Mensajes - Bean – Html

2.2 CLASES JAVA

2.2.1 Action

2.2.1.1EstadisticasAction.java

Descripción: Es nuestro Action Principal, el cual recibe como parámetros los criterios de consulta por parte del usuario, dependiendo de la opción escogida. Este Action se encarga de forma la ventana de la consulta dependiendo de la opción escogida, a su vez se encarga de hacer de revisar las sentencias sql considerando los criterios de la consulta solicitada. Cada consulta es enviada como un objeto el cual contiene la sentencia que se formada por los criterios escogidos por pantalla, el tipo de reporte a generar y la ventana de resultado.

2.2.2 Clases

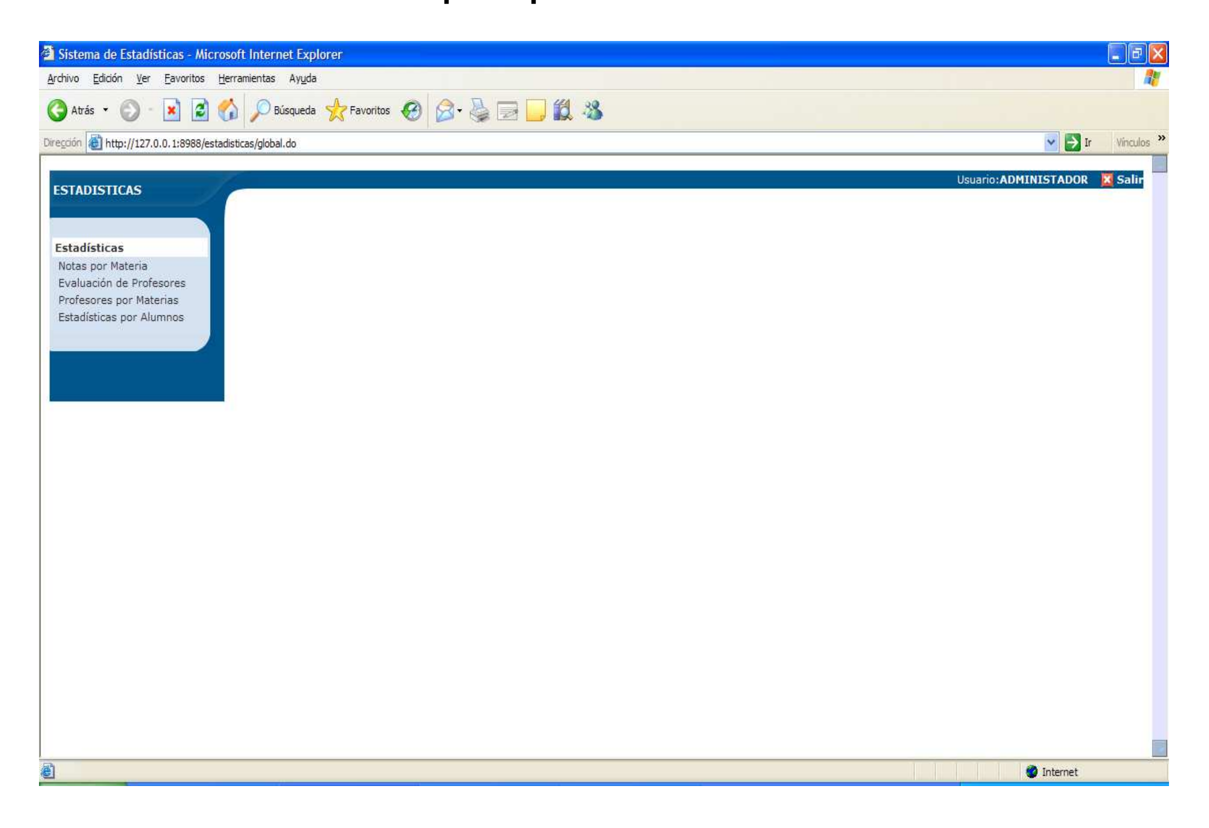
2.2.2.1 Alumnos

Se encarga de recuperar la información de la clase alumnos, de esta forma se puede conocer los datos personales de los alumnos y a su vez las materias y calificaciones que posee

3. DESCRIPCIÓN DE VENTANAS

3.1 Ventanas de consultas

3.1.1 Ventana principal de consulta



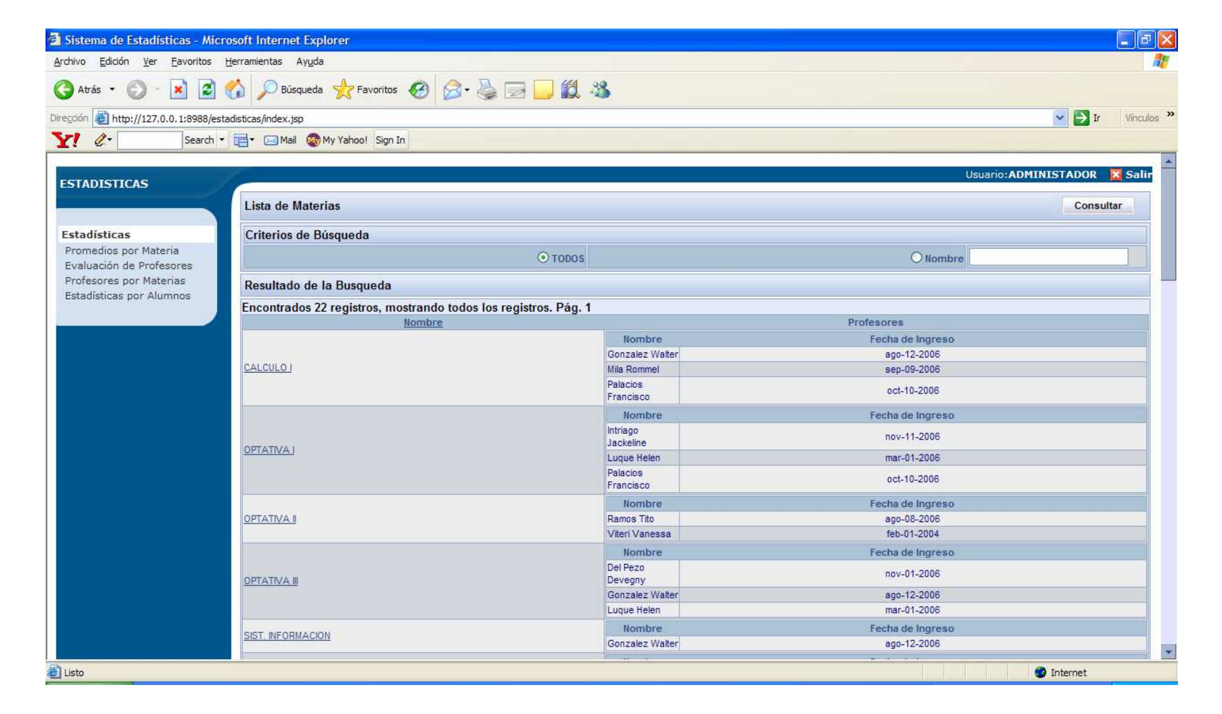
Descripción:

Esta pantalla contiene las opciones de consulta para la generación de reportes estadísticos, las alternativas son:

- **Profesores por materias:** En esta opción el usuario podrá visualizar el comportamiento de los alumnos (aprobados, reprobados y recuperación) en cada una de las materias que imparte el profesor que se ha seleccionado
- **Evaluación de profesores:** El usuario conocerá el resultado de las evaluaciones que se realizan durante un período para conocer el nivel de aceptación desde el punto de vista académico por parte de los alumnos de cada una de las materias impartidas por los docentes
- **Notas por materias:** Permite mostrar el comportamiento de los alumnos en las diferentes materias dictadas por varios docentes, lo que dará la posibilidad a conocer el rendimiento de los alumnos en una misma materia pero dictada por diferentes profesores.
- **Estadísticas de alumnos:** Esta opción permite conocer el rendimiento de un alumno en las diversas materias que cursa en un período de clases.
- **Las tablas consideradas para la generación de los gráficos se encuentran en el anexo A.**

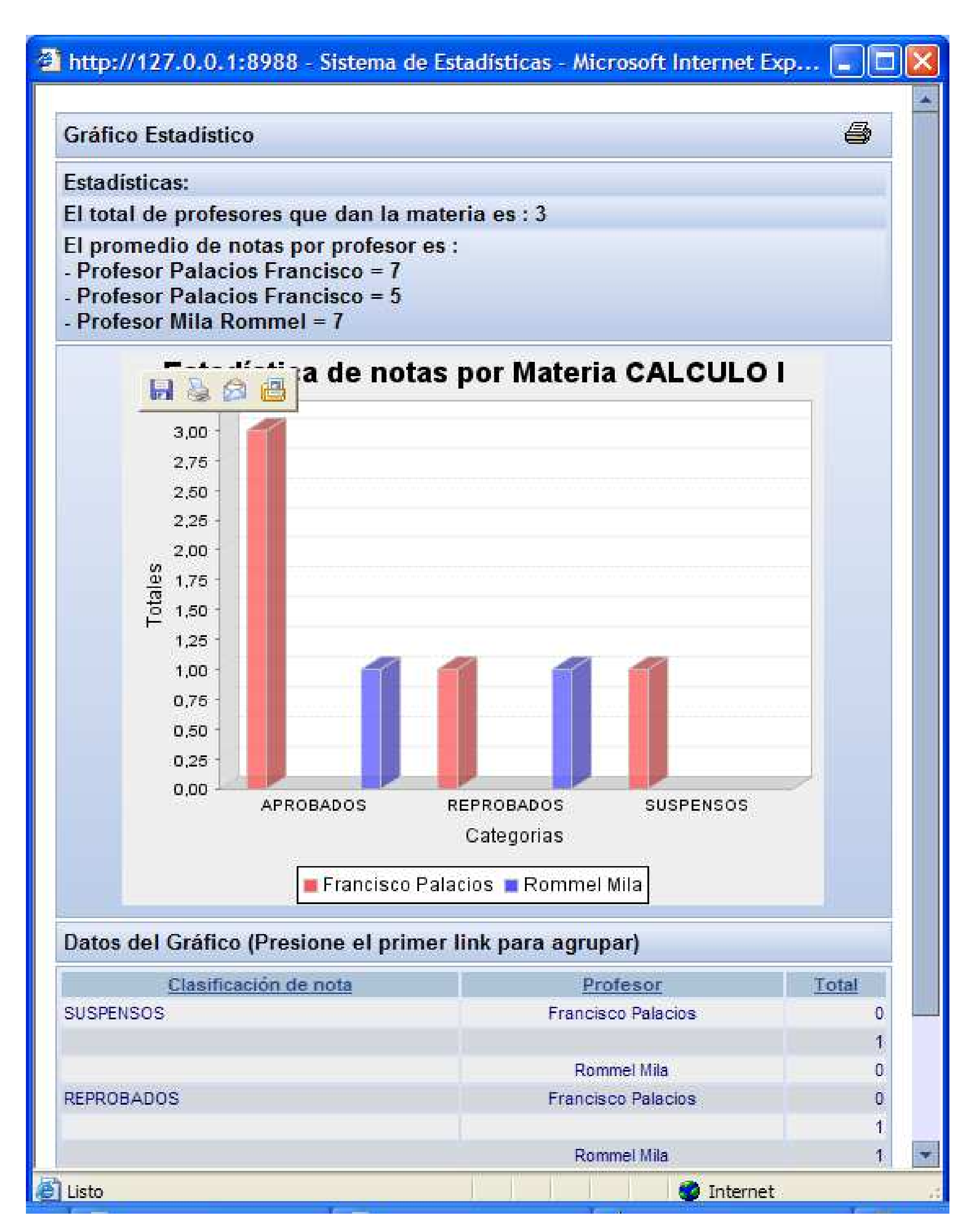
3.1.2 Notas por materias

Manteniendo el esquema original de las consultas, esta opción mostrará varias alternativas de consulta, se puede listar todas las materias o escoger el nombre de la materia. Luego de presionar el botón **consultar**, se muestra la cantidad de registros encontrados, el nombre del profesor, la fecha de ingreso, el año, período y las materias que dicta el docente.



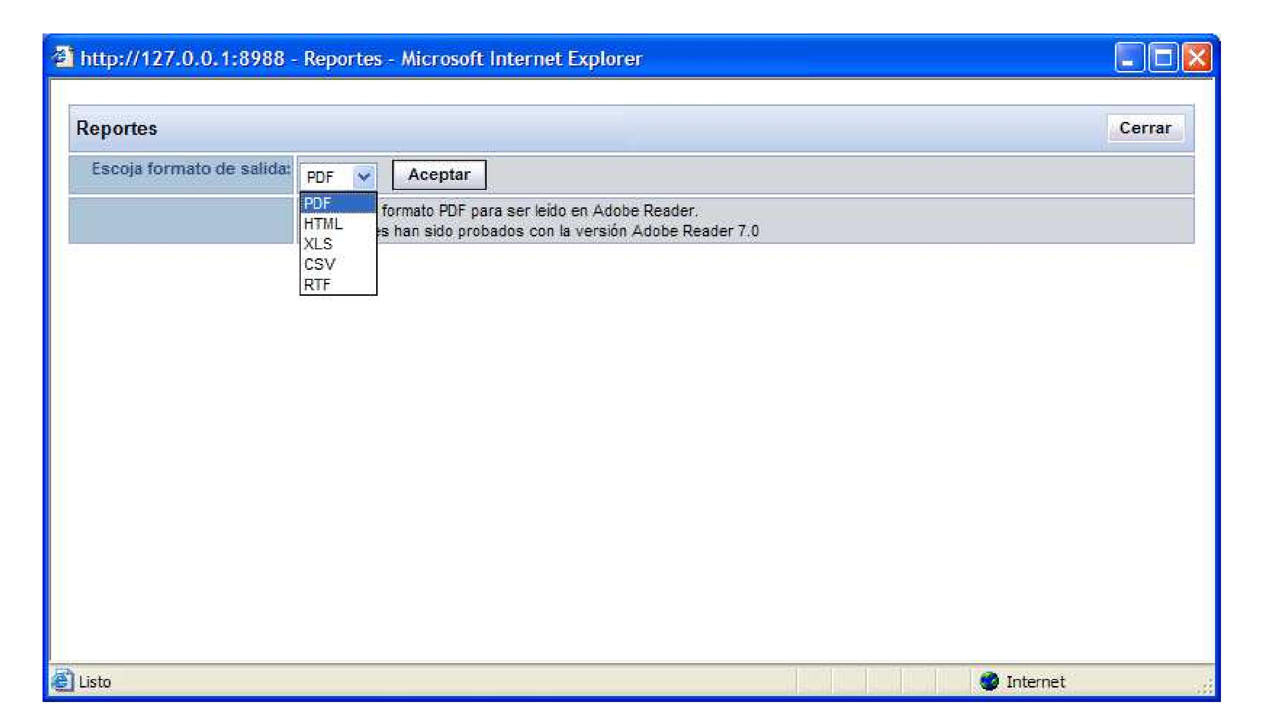
Si el usuario escoge una de las materias asignadas a un docente o varios docentes, el gráfico mostrará la cantidad de alumnos aprobados, reprobados y recuperación, en un gráfico de barra y una tabla resumida que contiene la

información solicitada.



Como se indica en la parte superior de la ventana, se encuentra expresada la cantidad de profesores que dan la materia en mención, cual es el promedio de cada profesor en los diferentes periodos que ha dado la materia.

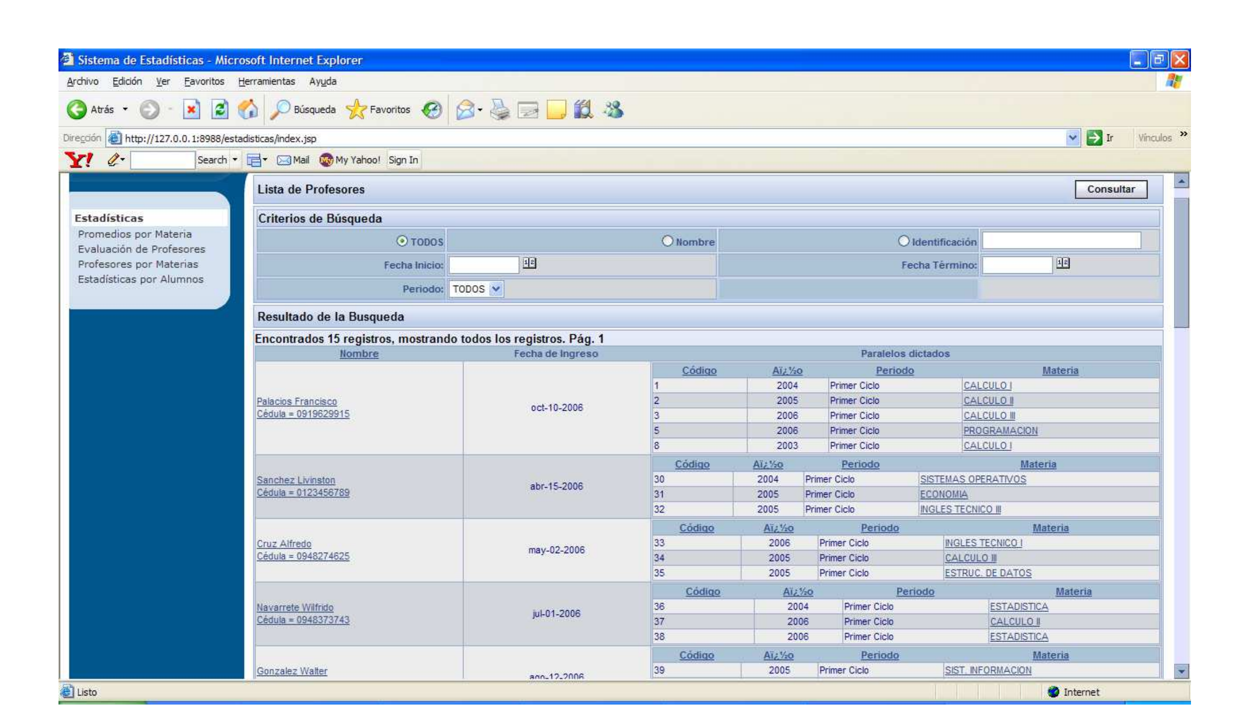
Cada ventana contiene el resultado de la consulta tiene la posibilidad de que las exportaciones de los datos sea de dos formas: Exportar los datos de las tablas a CSV (Texto separado por comas), Excel o XML. Otra alternativa es exportar toda la información, es decir, gráfico y tabla a los siguientes formatos: PDF, HTML, XLS, CSV, RTF (Texto enriquecido)



Nota.- Cabe indicar que por la versión de la herramienta generadora de reportes (1.2.0) los gráficos en Excel no pueden ser apreciados, solo las tablas, así como también, de forma esporádica, las barras de los gráficos tiende a cambiar si se hacen varios refrescamiento a las páginas en donde se encuentra los gráficos.

3.1.3 Profesores por materia

La ventana de Profesores por materias, da la posibilidad al usuario de conocer de forma gráfica el rendimiento de alumnos en la(s) materia(s) impartida por un profesor. El usuario puede listar todos los profesores y sus materias, puede ingresar el nombre del profesor o su identificación; así como también puede considerar la fecha de ingreso de los datos del profesor al sistema. Así como también podrá elegir que periodo desea evaluar.

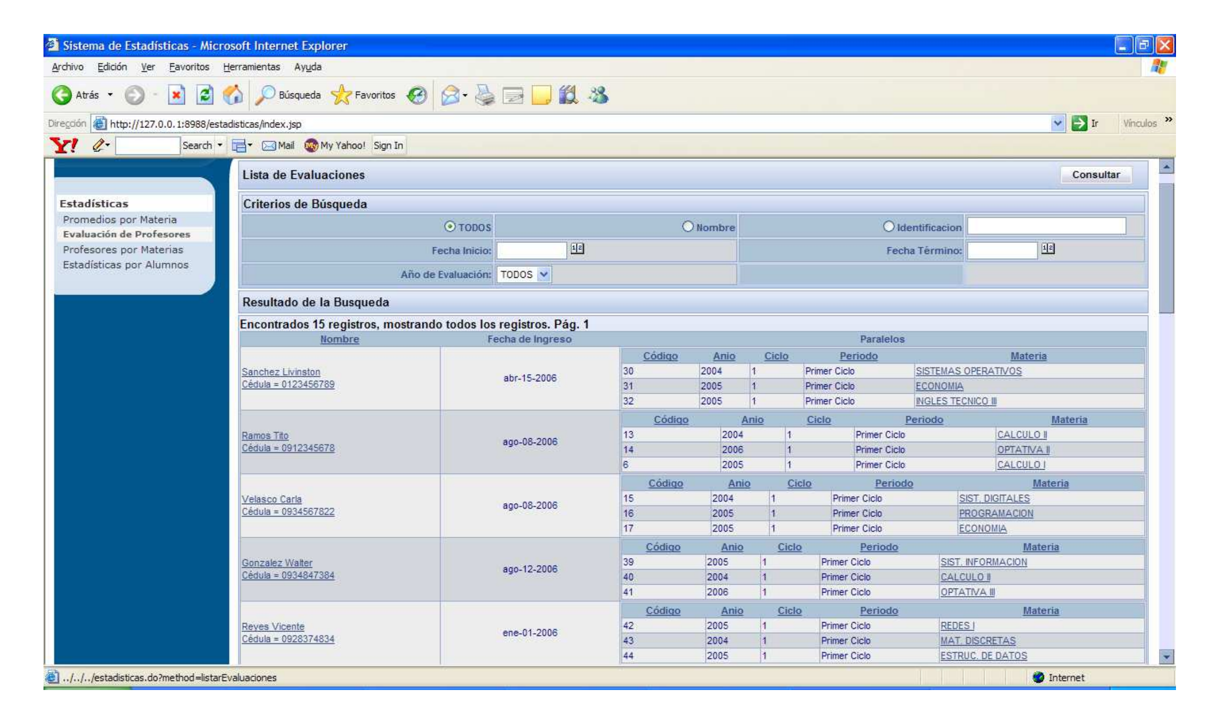


El resultado de la consulta será el siguiente: En la ventana se puede visualizar la cantidad de materias asignadas al profesor y cuales son las mismas, el total de alumnos evaluados, el promedio global


http://127.0.0.1:8988 - Sistema de Estadísticas - Microsoft Interne...
Profesor: Francisco Palacios
Materias Asigandas:
- CALCULO I
- PROGRAMACION
- CALCULO I
- CALCULO III
- CALCULO II
El total global de alumnos evaluados son : 11
El promedio global de la notas es : 7
CALCULO I. El total de almnos: 5. El promedio de las notas es: 5
PROGRAMACION. El total de almnos: 1. El promedio de las notas es: 10
CALCULO I. El total de almnos: 1. El promedio de las notas es: 7
CALCULO III. El total de almnos: 2. El promedio de las notas es: 9
CALCULO II. El total de almnos: 2. El promedio de las notas es: 6
Estadística de notas por Materia del profesor Francisco Palacios
Totales
2,00
1,75
1,50
1,25
1,00
0,75
0,50
0,25
0,00
APROBADOS
REPROBADOS
SUSPENSOS
Categorias
CALCULO I
PROGRAMACION
CALCULO III
CALCULO II
Datos del Gráfico (Presione el primer link para agrupar)
Clasificación de nota
Paralelo
Total
APROBADOS
CALCULO I
3
Listo
Internet

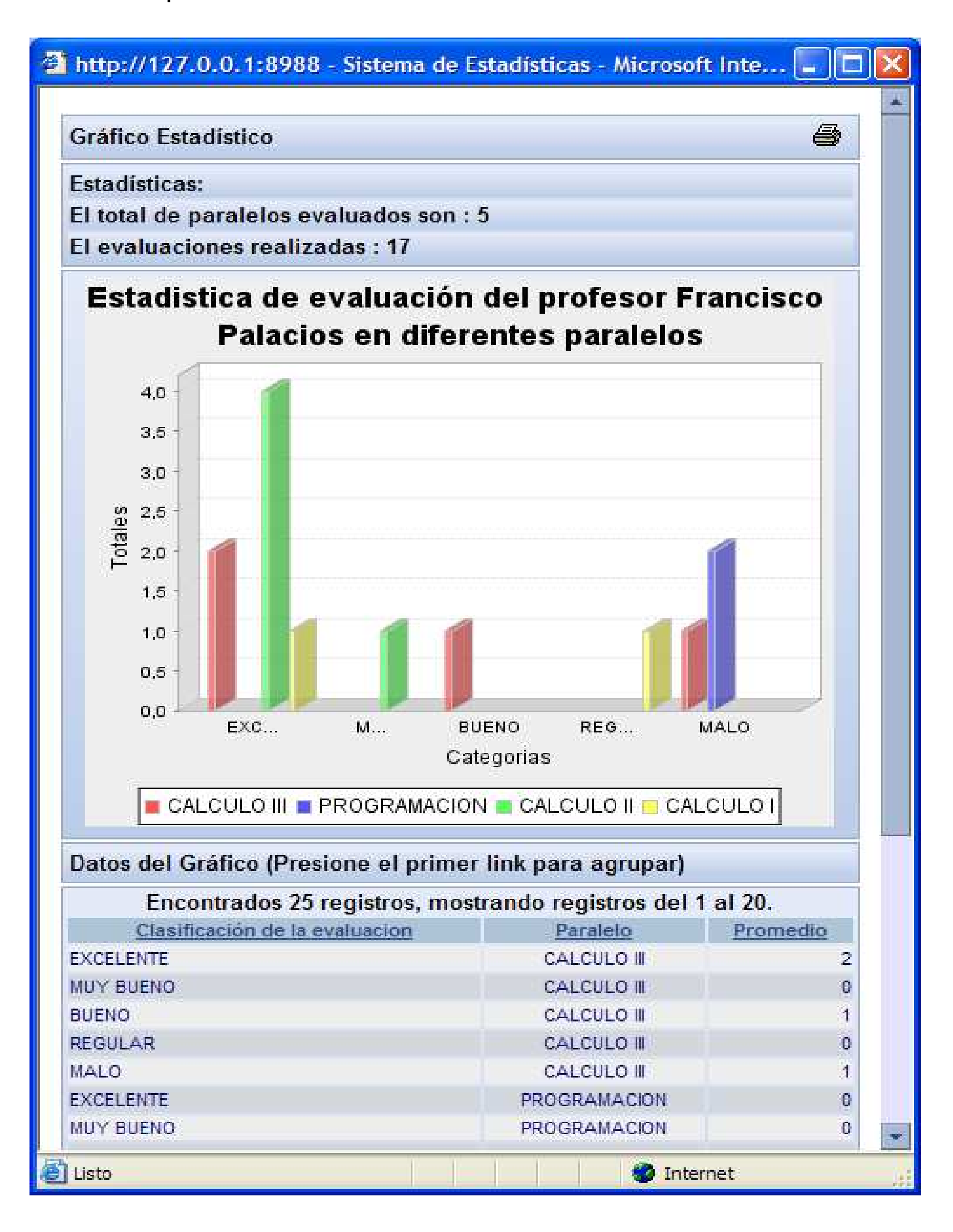
3.1.3 Evaluación de Profesores

Si el usuario desea visualizar la evaluación de los profesores, al momento de escoger dicha opción, podrá listar todos los profesores, buscar un profesor específico o ingresar su identificación; así como también se podrá escoger el periodo a evaluar.



El resultado, al escoger un profesor, es la visualización del nivel cualitativo en que las evaluaciones realizadas por los estudiantes durante un período de clases. Los parámetros de medición son: Excelente, muy bueno, bueno, regular y malo.

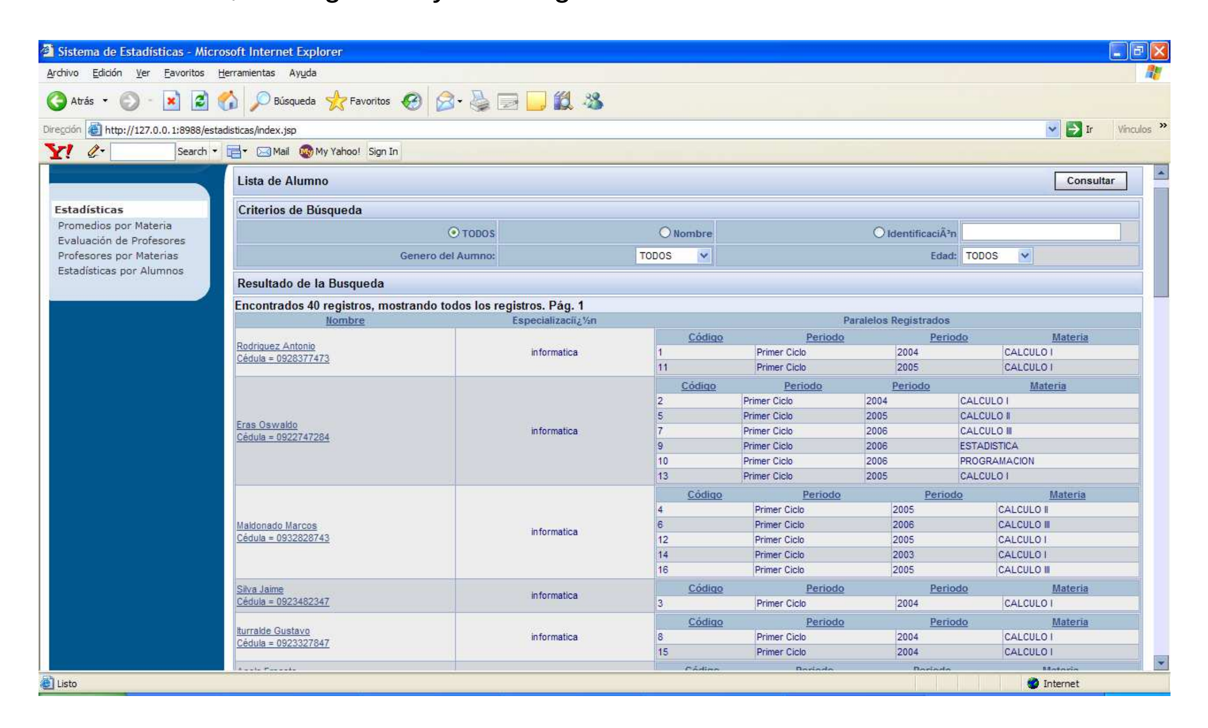
Como se puede apreciar en el gráfico, se muestra el total de paralelos evaluados, la cantidad de evaluaciones realizadas, así como cuales son las materias que han sido evaluadas.



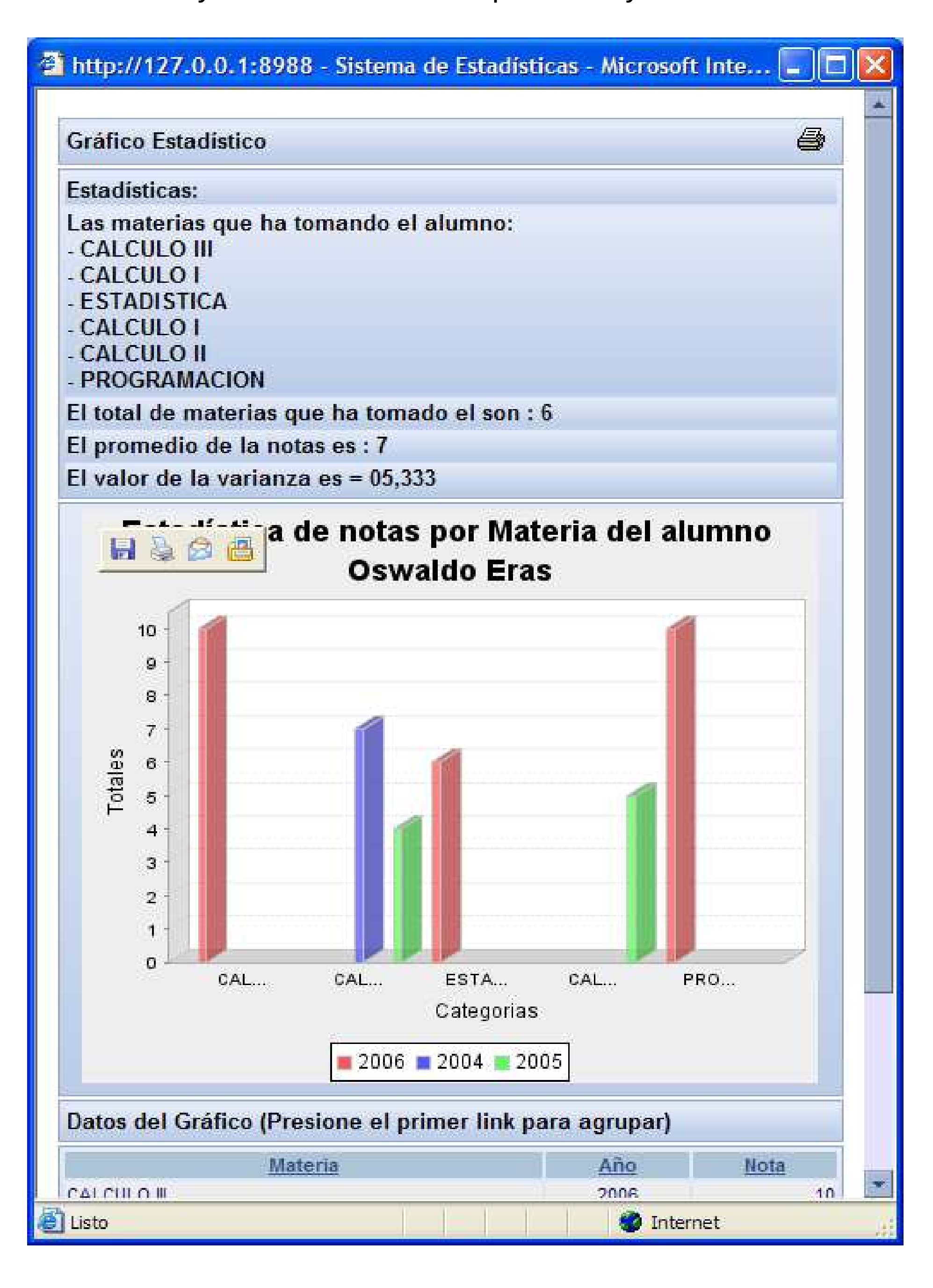
Clasificación de la evaluacion	Paralelo	Promedio
EXCELENTE	CALCULO III	2
MUY BUENO	CALCULO III	0
BUENO	CALCULO III	1
REGULAR	CALCULO III	0
MALO	CALCULO III	1
EXCELENTE	PROGRAMACION	0
MUY BUENO	PROGRAMACION	0

3.1.4 Estadísticas por Alumnos

En esta opción el usuario podrá listar todos los alumnos, se podrá listar todos, buscar por el nombre o por la identificación. Los criterios de selección considerados, es el género y los rangos de edad de los estudiantes



La finalidad de esta consulta es listar el rendimiento del alumno en las materias que ha tomado durante un período o periodos de clases. En este gráfico se muestra el total de materias tomadas por el alumno, el promedio de las notas y la varianza entre un promedio y otro.



MANUAL TÉCNICO

Clases

Alumnos.java

```
package com.bvg.comun.model.estadisticas;

import java.util.Date;
import java.util.Set;

import com.bvg.comun.persistencia.Entidad;
import com.bvg.comun.persistencia.EntidadEliminable;

public class Alumno extends Entidad {


	private Persona persona;
	private Especializacion especializacion;

	private Set registros;




	public Alumno() {
		super();
	}

	public Long getId() {
		return this.id;
	}


	public Persona getPersona() {
		return persona;
	}

	public void setPersona(Persona persona) {
		this.persona = persona;
	}
```

```
	public Especializacion getEspecializacion() {
		return especializacion;
	}
	public void setEspecializacion(Especializacion especializacion) {
		this.especializacion = especializacion;
	}

	public Set getRegistros() {
		return registros;
	}

	public void setRegistros(Set paralelos) {
		this.registros = paralelos;
	}


}
```

Profesor.java

```
package com.bvg.comun.model.estadisticas;


import java.util.Date;
import java.util.Set;

import com.bvg.comun.persistencia.EntidadEliminable;


public class Profesor extends EntidadEliminable {

	private Persona persona;
	private Set materias;
	private Set paralelos;

	private Date fechaIngreso;


	public Profesor() {
		super();
	}
```

```
	public Long getId() {
		return this.id;
	}


	public Persona getPersona() {
		return persona;
	}

	public void setPersona(Persona persona) {
		this.persona = persona;
	}


	public Set getMaterias() {
		return materias;
	}

	public void setMaterias(Set materias) {
		this.materias = materias;
	}


	public Date getFechaIngreso() {
		return fechaIngreso;
	}

	public void setFechaIngreso(Date fechaIngreso) {
		this.fechaIngreso = fechaIngreso;
	}


	public Set getParalelos() {
		return paralelos;
	}

	public void setParalelos(Set paralelos) {
		this.paralelos = paralelos;
	}


}
```

Materias.java

```
package com.bvg.comun.model.estadisticas;

import java.util.Set;

import com.bvg.comun.persistencia.Entidad;

public class Materia extends Entidad {


	private String nombre;
	private Set profesores;
	public Materia() {
		super();
	}

	public Long getId() {
		return this.id;
	}


	public String getNombre() {
		return nombre;
	}

	public void setNombre(String nombre) {
		this.nombre = nombre;
	}


	public Set getProfesores() {
		return profesores;
	}

	public void setProfesores(Set profesores) {
		this.profesores = profesores;
	}




}
```

Reporte.java

```
package com.bvg.comun.web.reporte;

import java.io.File;
import java.util.HashMap;
import java.util.List;

public class Reporte {
	public static final int TIPO_PDF=1;
	public static final int TIPO_HTML=2;
	public static final int TIPO_XLS=3;
	public static final int TIPO_CSV=4;
	public static final int TIPO_RTF=5;
	public static final int TIPO_XML=6;
	private String nombreReporteJasper;
	private String nombreReporteXML;
	private String nombreReporte;
	private String imagenDir;
	private File archivoReporteJasper = null;
	private List coleccionDatos = null;
	private HashMap parametros = new HashMap();

	private int tipoSalida;

	public String getNombreReporteJasper() {
		return nombreReporteJasper;
	}
	public void setNombreReporteJasper(String nombreReporteJasper) {
		this.nombreReporteJasper = nombreReporteJasper;
	}

	public String getNombreReporteXML() {
		return nombreReporteXML;
	}
	public void setNombreReporteXML(String nombreReporteXML) {
		this.nombreReporteXML = nombreReporteXML;
	}

	public String getNombreReporte() {
		return nombreReporte;
	}
	public void setNombreReporte(String nombreReporte) {
		this.nombreReporte = nombreReporte.trim();
```

```
        setNombreReporteXML(nombreReporte+".jrxml");
        setNombreReporteJasper(nombreReporte+".jasper");
}

public String getImagenDir() {
        return imagenDir;
}
public void setImagenDir(String imagenDir) {
        this.imagenDir = imagenDir;
}

public File getArchivoReporteJasper() {
        return archivoReporteJasper;
}
public void setArchivoReporteJasper(File archivoReporteJasper) {
        this.archivoReporteJasper = archivoReporteJasper;
}

public List getColeccionDatos() {
        return coleccionDatos;
}
public void setColeccionDatos(List coleccionDatos) {
        this.coleccionDatos = coleccionDatos;
}

public HashMap getParametros() {
        return parametros;
}
public void setParametros(HashMap parametros) {
        this.parametros = parametros;
}
public void addParametro(Object key, Object value){
        // Si ya existe, NO lo reemplaza
        if (parametros.get(key)==null){
                parametros.put(key,value);
        }
}


public int getTipoSalida() {
        return tipoSalida;
}
public void setTipoSalida(int tipoSalida) {
        this.tipoSalida = tipoSalida;
```

```
	}

	public String getTipoSalidaString() {
		switch (tipoSalida) {
		case TIPO_PDF:
			return "PDF";
		case TIPO_XLS:
			return "XLS";
		case TIPO_RTF:
			return "RTF";
		case TIPO_HTML:
			return "HTML";
		case TIPO_CSV:
			return "CSV";
		default:
			return "";
		}
	}
}
```

Action

EstadisticasAction.java

```
package com.bvg.app.web;

import java.io.OutputStream;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.DynaActionForm;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartUtilities;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.CategoryAxis;
```

```
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.renderer.category.BarRenderer;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.data.general.DefaultPieDataset;

import com.bvg.comun.model.estadisticas.Alumno;
import com.bvg.comun.model.estadisticas.Evaluacion;
import com.bvg.comun.model.estadisticas.Materia;
import com.bvg.comun.model.estadisticas.Paralelo;
import com.bvg.comun.model.estadisticas.Profesor;
import com.bvg.comun.model.estadisticas.Registro;
import com.bvg.comun.util.FechaFormat;
import com.bvg.comun.web.GlobalAction;
import com.bvg.comun.web.ReportesAction;

public class EstadisticasAction extends ReportesAction {

	//URL de la acción de struts
	public static final String SECURITY_ACCION = "estadisticas";


    public ActionForward listarAlumnos(ActionMapping mapping, ActionForm
form, HttpServletRequest request, HttpServletResponse response)
    throws Exception {
	String accion = (String) request.getSession().getAttribute(ACCION);
	if (accion == null || accion.length() <= 0){
		request.getSession().setAttribute(ACCION,
SECURITY_ACCION);
	}



		DynaActionForm dynaForm = (DynaActionForm) form;
		String textToFind = (String) dynaForm.get("textToFind");
		String cmbTextToFind = (String)
dynaForm.get("cmbTextToFind");

		List items = null;
		HashMap filtros = new HashMap();

		//List result = new ArrayList();

		if (cmbTextToFind != null && cmbTextToFind.length() > 0 ){
```

```
                        if (cmbTextToFind.compareTo("NOMBRE") == 0 &&
textToFind != null && textToFind.length() > 0){
                                filtros.put("persona.apellidos like
'"+textToFind.trim()+"%'",textToFind.trim()+"%");
                        }
                        if (cmbTextToFind.compareTo("IDENTIFICACION") == 0
&& textToFind != null && textToFind.length() > 0){
                                filtros.put("persona.cedula like
'"+textToFind.trim()+"%'",textToFind.trim()+"%");
                        }

                        items = manager.getAlumnos(filtros);


                }

                request.setAttribute("items", items);
        return mapping.findForward("listarAlumnos");



   }



   public ActionForward listarMaterias(ActionMapping mapping, ActionForm
form, HttpServletRequest request, HttpServletResponse response)
   throws Exception {
        String accion = (String) request.getSession().getAttribute(ACCION);
        if (accion == null || accion.length() <= 0){
                request.getSession().setAttribute(ACCION,
SECURITY_ACCION);
        }



                DynaActionForm dynaForm = (DynaActionForm) form;
                String textToFind = (String) dynaForm.get("textToFind");
                String cmbTextToFind = (String)
dynaForm.get("cmbTextToFind");
```

```
                List items = null;
                HashMap filtros = new HashMap();

                //List result = new ArrayList();

                if (cmbTextToFind != null && cmbTextToFind.length() > 0 ){
                        if (cmbTextToFind.compareTo("NOMBRE") == 0 &&
textToFind != null && textToFind.length() > 0){
                                filtros.put("persona.apellidos like
'"+textToFind.trim()+"%'",textToFind.trim()+"%");
                        }

                        items = manager.getMaterias(filtros);


                }

                request.setAttribute("items", items);
        return mapping.findForward("listarMaterias");



   }


   public ActionForward listarProfesores(ActionMapping mapping, ActionForm
form, HttpServletRequest request, HttpServletResponse response)
   throws Exception {
        String accion = (String) request.getSession().getAttribute(ACCION);
        if (accion == null || accion.length() <= 0){
                request.getSession().setAttribute(ACCION,
SECURITY_ACCION);
        }



                DynaActionForm dynaForm = (DynaActionForm) form;
                String textToFind = (String) dynaForm.get("textToFind");
                String cmbTextToFind = (String)
dynaForm.get("cmbTextToFind");
```

```
                String fecha_inicio = (String) dynaForm.get("fecha_inicio");
                String fecha_fin = (String) dynaForm.get("fecha_fin");



                List items = null;
                HashMap filtros = new HashMap();

                //List result = new ArrayList();

                if (cmbTextToFind != null && cmbTextToFind.length() > 0 ){
                        if (cmbTextToFind.compareTo("NOMBRE") == 0 &&
textToFind != null && textToFind.length() > 0){
                                filtros.put("persona.apellidos like
'"+textToFind.trim()+"%'",textToFind.trim()+"%");
                        }
                        if (cmbTextToFind.compareTo("IDENTIFICACION") == 0
&& textToFind != null && textToFind.length() > 0){
                                filtros.put("persona.cedula like
'"+textToFind.trim()+"%'",textToFind.trim()+"%");
                        }


                        if (fecha_inicio != null && fecha_inicio.length() > 0 ){
                                filtros.put("fechaIngreso >=
'"+fecha_inicio+"'",FechaFormat.toDate(fecha_inicio));

                        }

                        if (fecha_fin != null && fecha_fin.length() > 0 ){
                                filtros.put("fechaIngreso <=
'"+fecha_fin+"'",FechaFormat.toDate(fecha_fin));
                        }


                        items = manager.getProfesores(filtros);


                }

                request.setAttribute("items", items);
        return mapping.findForward("listarProfesores");
```

```
   }


   public ActionForward listarEvaluaciones(ActionMapping mapping,
ActionForm form, HttpServletRequest request, HttpServletResponse
response)
   throws Exception {
        String accion = (String) request.getSession().getAttribute(ACCION);
        if (accion == null || accion.length() <= 0){
                request.getSession().setAttribute(ACCION,
SECURITY_ACCION);
        }



                DynaActionForm dynaForm = (DynaActionForm) form;
                String textToFind = (String) dynaForm.get("textToFind");
                String cmbTextToFind = (String)
dynaForm.get("cmbTextToFind");



                String fecha_inicio = (String) dynaForm.get("fecha_inicio");
                String fecha_fin = (String) dynaForm.get("fecha_fin");



                List items = null;
                HashMap filtros = new HashMap();

                //List result = new ArrayList();

                if (cmbTextToFind != null && cmbTextToFind.length() > 0 ){
                        if (cmbTextToFind.compareTo("NOMBRE") == 0 &&
textToFind != null && textToFind.length() > 0){
                                filtros.put("persona.apellidos like
'"+textToFind.trim()+"%'",textToFind.trim()+"%");
                        }
                        if (cmbTextToFind.compareTo("IDENTIFICACION") == 0
&& textToFind != null && textToFind.length() > 0){
                                filtros.put("persona.cedula like
'"+textToFind.trim()+"%'",textToFind.trim()+"%");
```

```
                        }


                        if (fecha_inicio != null && fecha_inicio.length() > 0 ){
                                filtros.put("fechaIngreso >=
'"+fecha_inicio+"'",FechaFormat.toDate(fecha_inicio));

                        }

                        if (fecha_fin != null && fecha_fin.length() > 0 ){
                                filtros.put("fechaIngreso <=
'"+fecha_fin+"'",FechaFormat.toDate(fecha_fin));
                        }


                        items = manager.getProfesores(filtros);


                }

                request.setAttribute("items", items);

        return mapping.findForward("listarEvaluaciones");



   }


   public ActionForward mostrarImageneEstadistica(ActionMapping mapping,
ActionForm form, HttpServletRequest request, HttpServletResponse
response)
   throws Exception {
           String titulo = (String) request.getSession().getAttribute("titulo");
           if (titulo != null ){
                // ignore
           }else{
                titulo = "Gráfico Estadístico";
           }
           List datosGraficoPastel = (List)
request.getSession().getAttribute("datosGraficoPastel");
           JFreeChart chart = createPieChart(titulo,  datosGraficoPastel);
      if (chart != null) {
```

```
          response.addHeader("Content-disposition", "attachment;
filename=imagen.png") ;
          response.setContentType("image/gif, image/x-xbitmap, image/jpeg,
image/pjpeg, image/jpg, image/jpe, image/png");
          OutputStream stream = response.getOutputStream();

          ChartUtilities.writeChartAsPNG(stream, chart, 430, 350);
       }
       return null;
   }

   /**
    * Creates a sample pie chart.
    *
    * @return a pie chart.
    */
    private JFreeChart createPieChart(String title, List tabla) {
        DefaultPieDataset data = new DefaultPieDataset();
        DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        int count = 1;
        for (Iterator iter = tabla.iterator(); iter.hasNext();) {
                        Object[] list = (Object[]) iter.next();


                        if (list.length == 2){

         dataset.addValue(Double.parseDouble(list[1].toString()),"",
list[0].toString() );
                        }else if (list.length == 3){

         dataset.addValue(Double.parseDouble(list[2].toString()),list[1].toString(
),list[0].toString());
                        }
                }


        JFreeChart chart2 = ChartFactory.createBarChart3D(
                        title, // chart title
                "Categorias", // domain axis label
                "Totales", // range axis label
                dataset, // data
                PlotOrientation.VERTICAL,
                true, // include legend
                true, // tooltips?
```

```
                false // URLs?
                );

            CategoryPlot plot = chart2.getCategoryPlot();
            CategoryAxis axis = plot.getDomainAxis();
            //axis.setLowerMargin(0.02); // two percent
            axis.setCategoryMargin(0.20); // ten percent
            //axis.setUpperMargin(0.02); // two percent

            BarRenderer renderer = (BarRenderer) plot.getRenderer();
            renderer.setItemMargin(0.50); // fifteen percent

            return chart2;
        }


        public ActionForward generarEstadisticaMateria(ActionMapping mapping,
ActionForm form, HttpServletRequest request, HttpServletResponse
response)
        throws Exception {
            String accion = (String) request.getSession().getAttribute(ACCION);
            if (accion == null || accion.length() <= 0){
                request.getSession().setAttribute(ACCION,
SECURITY_ACCION);
            }


            Profesor profesor = null;

                String id_paralelo = (String)
request.getParameter("id_profesor");
                if (id_paralelo != null && id_paralelo.length() > 0  ){
                    HashMap filtros = new HashMap();
                    filtros.put("id = '"+Long.valueOf(id_paralelo)+"'", null);
                    List profesores = manager.getProfesores(filtros);

                    if (profesores != null && profesores.size() >0 ){
                        profesor = (Profesor) profesores.get(0);
                        List datosGrafico = new ArrayList();
                        for (Iterator iter = profesor.getParalelos().iterator();
iter.hasNext();) {
                            Paralelo paralelo2 = (Paralelo) iter.next();

                            filtros.clear();
```

```
					filtros.put("id = '"+paralelo2.getId()+"'", null);
					List paralelos =
manager.getParalelos(filtros);

					if (paralelos != null && paralelos.size() > 0){
						Paralelo paralelo = (Paralelo)
paralelos.get(0);

						int aprobados = 0;
						int reprobados = 0;
						int suspensos = 0;

						for (Iterator iter4 =
paralelo.getRegistros().iterator(); iter4.hasNext();) {
							Registro registro = (Registro)
iter4.next();
							if
(registro.getPromedio().intValue() >= 7){
								aprobados++;
							}else if
(registro.getPromedio().intValue() >= 5 && registro.getPromedio().intValue() <
7){
								suspensos++;
							}else{
								reprobados++;
							}
						}


						datosGrafico.add(new
Object[]{"APROBADOS",
paralelo2.getMateria().getNombre(),Integer.valueOf(aprobados)});
						datosGrafico.add(new
Object[]{"REPROBADOS",paralelo2.getMateria().getNombre(),
Integer.valueOf(reprobados)});
						datosGrafico.add(new
Object[]{"SUSPENSOS",paralelo2.getMateria().getNombre(),
Integer.valueOf(suspensos)});



					//List datosGrafico =
manager.getEstadisticasPersonalByParaleloNota(id_linea_especialidad);
```

```
                                    }

                            }

        request.getSession().setAttribute("datosGraficoPastel", datosGrafico);

                                    //request.getSession().setAttribute("total",
Integer.valueOf(aprobados+reprobados+suspensos));

                        }

                    }
                    String nombre
=(profesor!=null)?(profesor.getPersona().getApellidos()+profesor.getPersona(
).getApellidos()):"";
                    request.getSession().setAttribute("titulo", "Clasificación de notas
por Materia del profesor " + nombre );
                    request.setAttribute("urlAccion", "generarEstadisticaMateria");
            return mapping.findForward("mostrarEstadisticas");

    }

    public ActionForward
generarEstadisticaMateriaConProfesores(ActionMapping mapping,
ActionForm form, HttpServletRequest request, HttpServletResponse
response)
    throws Exception {
        String accion = (String) request.getSession().getAttribute(ACCION);
        if (accion == null || accion.length() <= 0){
```

```
                request.getSession().setAttribute(ACCION,
SECURITY_ACCION);
        }


        List datosGrafico = new ArrayList();
        Materia materia = null;
                String id_paralelo = (String) request.getParameter("id_materia");
                if (id_paralelo != null && id_paralelo.length() > 0  ){
                        HashMap filtros = new HashMap();
                        filtros.put("id = '"+Long.valueOf(id_paralelo)+"'", null);
                        List maaterias = manager.getMaterias(filtros);

                        if (maaterias != null && maaterias.size() >0 ){
                                materia = (Materia) maaterias.get(0);

                                for (Iterator iter = materia.getProfesores().iterator();
iter.hasNext();) {
                                        Profesor profesor = (Profesor) iter.next();

                                        filtros.clear();
                                        filtros.put("id = '"+profesor.getId()+"'", null);

                                        List profesores =
manager.getProfesores(filtros);

                                        if(profesores != null && profesores.size() > 0
){
                                                Profesor profesor2 = (Profesor)
profesores.get(0);


                                                for (Iterator iterator =
profesor2.getParalelos().iterator(); iterator
                                                .hasNext();) {


                                                        Paralelo paralelo2 = (Paralelo)
iterator.next();


                                                        filtros.clear();
                                                        filtros.put("id =
'"+paralelo2.getId()+"'", null);
```

```
filtros.put("materia.id =
'"+materia.getId()+"'", null);
List paralelos =
manager.getParalelos(filtros);

if (paralelos != null &&
paralelos.size() > 0){
Paralelo paralelo =
(Paralelo) paralelos.get(0);

int aprobados = 0;
int reprobados = 0;
int suspensos = 0;

for (Iterator iter4 =
paralelo.getRegistros().iterator(); iter4.hasNext();) {
Registro registro
= (Registro) iter4.next();
if
(registro.getPromedio().intValue() >= 7){

aprobados++;
}else if
(registro.getPromedio().intValue() >= 5 && registro.getPromedio().intValue() <
7){

suspensos++;
}else{

reprobados++;
}
}


datosGrafico.add(new
Object[]{"APROBADOS", profesor2.getPersona().getNombres()+"
"+profesor2.getPersona().getApellidos(),Integer.valueOf(aprobados)});
datosGrafico.add(new
Object[]{"REPROBADOS",profesor2.getPersona().getNombres()+"
"+profesor2.getPersona().getApellidos(), Integer.valueOf(reprobados)});
datosGrafico.add(new
Object[]{"SUSPENSOS",profesor2.getPersona().getNombres()+"
"+profesor2.getPersona().getApellidos(), Integer.valueOf(suspensos)});
```

```
                                                                    //List datosGrafico =
manager.getEstadisticasPersonalByParaleloNota(id_linea_especialidad);

                                                                }
                                                        }
                                                }
                                        }
                                }
                        }

                        request.getSession().setAttribute("datosGraficoPastel",
datosGrafico);
                        request.getSession().setAttribute("tipo3", new
String[]{"Clasificación de nota", "Profesor", "Total"});

                        String nombre =(materia!=null)?(materia.getNombre()):"";
                        request.getSession().setAttribute("titulo", "Estadística de notas
por Materia " + nombre );

                        request.setAttribute("urlAccion",
"generarEstadisticaMateriaConProfesores");
                return mapping.findForward("mostrarEstadisticas");

    }



    public ActionForward generarEstadisticaAlumno(ActionMapping mapping,
ActionForm form, HttpServletRequest request, HttpServletResponse
response)
    throws Exception {
        String accion = (String) request.getSession().getAttribute(ACCION);
        if (accion == null || accion.length() <= 0){
                request.getSession().setAttribute(ACCION,
SECURITY_ACCION);
        }


        Alumno alumno = null;

                String id_alumno = (String) request.getParameter("id_alumno");
                if (id_alumno != null && id_alumno.length() > 0  ){
```

```
HashMap filtros = new HashMap();
filtros.put("id = '"+Long.valueOf(id_alumno)+"'", null);
List profesores = manager.getAlumnos(filtros);

if (profesores != null && profesores.size() >0 ){
    alumno = (Alumno) profesores.get(0);
    List datosGrafico = new ArrayList();
        int aprobados = 0;
        int reprobados = 0;
        int suspensos = 0;

    for (Iterator iter = alumno.getRegistros().iterator();
iter.hasNext();) {
        Registro registro = (Registro) iter.next();

                if
(registro.getPromedio().intValue() >= 7){
                    aprobados++;
                }else if
(registro.getPromedio().intValue() >= 5 && registro.getPromedio().intValue() <
7){
                    suspensos++;
                }else{
                    reprobados++;
                }

            datosGrafico.add(new
Object[]{"APROBADOS",
registro.getParalelo().getMateria().getNombre(),Integer.valueOf(aprobados)});
            datosGrafico.add(new
Object[]{"REPROBADOS",registro.getParalelo().getMateria().getNombre(),
Integer.valueOf(reprobados)});
            datosGrafico.add(new
Object[]{"SUSPENSOS",registro.getParalelo().getMateria().getNombre(),
Integer.valueOf(suspensos)});
```

```
                                        //List datosGrafico =
manager.getEstadisticasPersonalByParaleloNota(id_linea_especialidad);


                              }


        request.getSession().setAttribute("datosGraficoPastel", datosGrafico);

                                     request.getSession().setAttribute("tipo3",
new String[]{"Clasificación de nota", "Paralelo", "Total"});
                                     //request.getSession().setAttribute("total",
Integer.valueOf(aprobados+reprobados+suspensos));




                       }


                }
                String nombre
=(alumno!=null)?(alumno.getPersona().getNombres()+"
"+alumno.getPersona().getApellidos()):"";
                request.getSession().setAttribute("titulo", "Estadística de notas
por Materia del alumno " + nombre );


                request.setAttribute("urlAccion", "generarEstadisticaAlumno");
           return mapping.findForward("mostrarEstadisticas");

    }


    public ActionForward generarEstadisticaProfesor(ActionMapping mapping,
ActionForm form, HttpServletRequest request, HttpServletResponse
response)
    throws Exception {
        String accion = (String) request.getSession().getAttribute(ACCION);
        if (accion == null || accion.length() <= 0){
```

```
                request.getSession().setAttribute(ACCION,
SECURITY_ACCION);
        }


        Profesor profesor = null;

                String id_paralelo = (String)
request.getParameter("id_profesor");
                if (id_paralelo != null && id_paralelo.length() > 0  ){
                        HashMap filtros = new HashMap();
                        filtros.put("id = '"+Long.valueOf(id_paralelo)+"'", null);
                        List profesores = manager.getProfesores(filtros);

                        if (profesores != null && profesores.size() >0 ){
                                profesor = (Profesor) profesores.get(0);
                                List datosGrafico = new ArrayList();
                                for (Iterator iter = profesor.getParalelos().iterator();
iter.hasNext();) {
                                        Paralelo paralelo2 = (Paralelo) iter.next();

                                        filtros.clear();
                                        filtros.put("id = '"+paralelo2.getId()+"'", null);
                                        List paralelos =
manager.getParalelos(filtros);

                                        if (paralelos != null && paralelos.size() > 0){
                                                Paralelo paralelo = (Paralelo)
paralelos.get(0);

                                                int aprobados = 0;
                                                int reprobados = 0;
                                                int suspensos = 0;

                                                for (Iterator iter4 =
paralelo.getRegistros().iterator(); iter4.hasNext();) {
                                                        Registro registro = (Registro)
iter4.next();
                                                        if
(registro.getPromedio().intValue() >= 7){
                                                                aprobados++;
                                                        }else if
(registro.getPromedio().intValue() >= 5 && registro.getPromedio().intValue() <
7){
```

```
                                        suspensos++;
                                }else{
                                        reprobados++;
                                }
                        }


                        datosGrafico.add(new
Object[]{"APROBADOS",
paralelo2.getMateria().getNombre(),Integer.valueOf(aprobados)});
                        datosGrafico.add(new
Object[]{"REPROBADOS",paralelo2.getMateria().getNombre(),
Integer.valueOf(reprobados)});
                        datosGrafico.add(new
Object[]{"SUSPENSOS",paralelo2.getMateria().getNombre(),
Integer.valueOf(suspensos)});



                    //List datosGrafico =
manager.getEstadisticasPersonalByParaleloNota(id_linea_especialidad);

                }



            }


    request.getSession().setAttribute("datosGraficoPastel", datosGrafico);

                request.getSession().setAttribute("tipo3",
new String[]{"Clasificación de nota", "Paralelo", "Total"});
                //request.getSession().setAttribute("total",
Integer.valueOf(aprobados+reprobados+suspensos));





        }


    }
```

```
                String nombre
=(profesor!=null)?(profesor.getPersona().getNombres()+"
"+profesor.getPersona().getApellidos()):"";
                request.getSession().setAttribute("titulo", "Estadística de notas
por Materia del profesor " + nombre );


                request.setAttribute("urlAccion", "generarEstadisticaProfesor");
           return mapping.findForward("mostrarEstadisticas");

    }

    public ActionForward
generarEstadisticaEvaluacionProfesor(ActionMapping mapping, ActionForm
form, HttpServletRequest request, HttpServletResponse response)
    throws Exception {
        String accion = (String) request.getSession().getAttribute(ACCION);
        if (accion == null || accion.length() <= 0){
                request.getSession().setAttribute(ACCION,
SECURITY_ACCION);
        }


        Profesor profesor = null;

                String id_paralelo = (String)
request.getParameter("id_profesor");
                if (id_paralelo != null && id_paralelo.length() > 0  ){
                        HashMap filtros = new HashMap();
                        filtros.put("id = '"+Long.valueOf(id_paralelo)+"'", null);
                        List profesores = manager.getProfesores(filtros);

                        if (profesores != null && profesores.size() >0 ){
                                profesor = (Profesor) profesores.get(0);
                                List datosGrafico = new ArrayList();
                                for (Iterator iter = profesor.getParalelos().iterator();
iter.hasNext();) {
                                        Paralelo paralelo2 = (Paralelo) iter.next();

                                        filtros.clear();
                                        filtros.put("id = '"+paralelo2.getId()+"'", null);
                                        List paralelos =
manager.getParalelosEvaluacion(filtros);
```

```
if (paralelos != null && paralelos.size() > 0){
        Paralelo paralelo = (Paralelo)
paralelos.get(0);

        int excenlente = 0;
        int muy_bueno = 0;
        int bueno = 0;
        int regular = 0;
        int malo = 0;


        for (Iterator iter2 =
paralelo.getEvaluaciones().iterator(); iter2.hasNext();) {
                Evaluacion evaluacion =
(Evaluacion) iter2.next();
                if
(evaluacion.getRespuesta().intValue() ==5){
                        excenlente++;
                }else if
(evaluacion.getRespuesta().intValue() == 4){
                        muy_bueno++;
                }else if
(evaluacion.getRespuesta().intValue() == 3){
                        bueno++;
                }else if
(evaluacion.getRespuesta().intValue() == 2){
                        regular++;
                }else if
(evaluacion.getRespuesta().intValue() == 1){
                        malo++;
                }
        }



        datosGrafico.add(new
Object[]{"EXCELENTE", paralelo2.getMateria().getNombre(),
Integer.valueOf(excenlente)});
        datosGrafico.add(new Object[]{"MUY
BUENO", paralelo2.getMateria().getNombre(), Integer.valueOf(muy_bueno)});
        datosGrafico.add(new
Object[]{"BUENO", paralelo2.getMateria().getNombre(),
Integer.valueOf(bueno)});
```

```
                                        datosGrafico.add(new
Object[]{"REGULAR", paralelo2.getMateria().getNombre(),
Integer.valueOf(regular)});
                                        datosGrafico.add(new
Object[]{"MALO", paralelo2.getMateria().getNombre(),
Integer.valueOf(malo)});



                                    //List datosGrafico =
manager.getEstadisticasPersonalByParaleloNota(id_linea_especialidad);

                                  }



                           }


       request.getSession().setAttribute("datosGraficoPastel", datosGrafico);


                           request.getSession().setAttribute("tipo3", new
String[]{"Clasificación de la evaluacion", "Paralelo","Promedio"});





                    }


             }

             String nombre
=(profesor!=null)?(profesor.getPersona().getNombres()+"
"+profesor.getPersona().getApellidos()):"";
             request.getSession().setAttribute("titulo", "Estadistica de
evaluación del profesor " + nombre  + " en diferentes paralelos");


             request.setAttribute("urlAccion",
"generarEstadisticaEvaluacionProfesor");
          return mapping.findForward("mostrarEstadisticas");
```

```
    }

    public ActionForward
generarEstadisticaEvaluacionParalelo(ActionMapping mapping, ActionForm
form, HttpServletRequest request, HttpServletResponse response)
    throws Exception {
        String accion = (String) request.getSession().getAttribute(ACCION);
        if (accion == null || accion.length() <= 0){
                request.getSession().setAttribute(ACCION,
SECURITY_ACCION);
        }

        Paralelo paralelo = null;
                String id_paralelo = (String)
request.getParameter("id_paralelo");
                if (id_paralelo != null && id_paralelo.length() > 0  ){
                        HashMap filtros = new HashMap();
                        filtros.put("id = '"+id_paralelo+"'", null);
                        List paralelos = manager.getParalelosEvaluacion(filtros);

                        if (paralelos != null && paralelos.size() > 0){
                                paralelo = (Paralelo) paralelos.get(0);

                                int excenlente = 0;
                                int muy_bueno = 0;
                                int bueno = 0;
                                int regular = 0;
                                int malo = 0;


                                for (Iterator iter =
paralelo.getEvaluaciones().iterator(); iter.hasNext();) {
                                        Evaluacion evaluacion = (Evaluacion)
iter.next();
                                        if (evaluacion.getRespuesta().intValue()
==5){
                                                excenlente++;
                                        }else if
(evaluacion.getRespuesta().intValue() == 4){
                                                muy_bueno++;
                                        }else if
(evaluacion.getRespuesta().intValue() == 3){
                                                bueno++;
```

```
                    }else if
(evaluacion.getRespuesta().intValue() == 2){
                        regular++;
                    }else if
(evaluacion.getRespuesta().intValue() == 1){
                        malo++;
                    }
                }

                List datosGrafico = new ArrayList();
                datosGrafico.add(new Object[]{"EXCELENTE",
Integer.valueOf(excenlente)});
                datosGrafico.add(new Object[]{"MUY BUENO",
Integer.valueOf(muy_bueno)});
                datosGrafico.add(new Object[]{"BUENO",
Integer.valueOf(bueno)});
                datosGrafico.add(new Object[]{"REGULAR",
Integer.valueOf(regular)});
                datosGrafico.add(new Object[]{"MALO",
Integer.valueOf(malo)});



    request.getSession().setAttribute("datosGraficoPastel", datosGrafico);
                request.getSession().setAttribute("total",
Integer.valueOf((excenlente+muy_bueno+bueno+regular+malo)/2));

                request.getSession().setAttribute("tipo2", new
String[]{"Clasificación de la evaluacion", "Promedio"});

              //List datosGrafico =
manager.getEstadisticasPersonalByParaleloNota(id_linea_especialidad);

            }

        }

        String nombre =(paralelo!=null)?(paralelo.getId()):"";
        String profesor
=(paralelo!=null)?(paralelo.getProfesor().getPersona().getNombres()+paralelo
.getProfesor().getPersona().getApellidos()):"";
        request.getSession().setAttribute("titulo", "Estadistica de
evaluación del  " + profesor + " en el paralelo " + nombre);
```

```
                request.setAttribute("urlAccion",
"generarEstadisticaEvaluacionParalelo");
            return mapping.findForward("mostrarEstadisticas");

    }



    public ActionForward generarEstadisticaParaleloMateria(ActionMapping
mapping, ActionForm form, HttpServletRequest request,
HttpServletResponse response)
    throws Exception {
        String accion = (String) request.getSession().getAttribute(ACCION);
        if (accion == null || accion.length() <= 0){
                request.getSession().setAttribute(ACCION,
SECURITY_ACCION);
        }
        Paralelo paralelo = null;

                String id_paralelo = (String)
request.getParameter("id_paralelo");
                if (id_paralelo != null && id_paralelo.length() > 0  ){

                        HashMap filtros = new HashMap();
                        filtros.put("id = '"+id_paralelo+"'", null);
                        List paralelos = manager.getParalelos(filtros);

                        if (paralelos != null && paralelos.size() > 0){
                                paralelo = (Paralelo) paralelos.get(0);

                                int aprobados = 0;
                                int reprobados = 0;
                                int suspensos = 0;

                                for (Iterator iter = paralelo.getRegistros().iterator();
iter.hasNext();) {
                                        Registro registro = (Registro) iter.next();
                                        if (registro.getPromedio().intValue() >= 7){
                                                aprobados++;
                                        }else if (registro.getPromedio().intValue() >=
5 && registro.getPromedio().intValue() < 7){
                                                suspensos++;
                                        }else{
```

```
                                        reprobados++;
                                    }
                                }

                                List datosGrafico = new ArrayList();
                                datosGrafico.add(new Object[]{"APROBADOS",
Integer.valueOf(aprobados)});
                                datosGrafico.add(new Object[]{"REPROBADOS",
Integer.valueOf(reprobados)});
                                datosGrafico.add(new Object[]{"SUSPENSOS",
Integer.valueOf(suspensos)});


        request.getSession().setAttribute("datosGraficoPastel", datosGrafico);
                                request.getSession().setAttribute("total",
Integer.valueOf(aprobados+reprobados+suspensos));

                                request.getSession().setAttribute("tipo2", new
String[]{"Clasificación de nota", "Total"});

                             //List datosGrafico =
manager.getEstadisticasPersonalByParaleloNota(id_linea_especialidad);

                         }

                }

                String nombre =(paralelo!=null)?(paralelo.getId()):"";
                String profesor
=(paralelo!=null)?(paralelo.getProfesor().getPersona().getNombres()+"
"+paralelo.getProfesor().getPersona().getApellidos()):"";
                request.getSession().setAttribute("titulo", "Estadistica de las
notas del  profesor " + profesor + " en el paralelo " + nombre);

                request.setAttribute("urlAccion",
"generarEstadisticaParaleloMateria");
           return mapping.findForward("mostrarEstadisticas");

   }



}
```